Comparison of Linked Lists and Dynamic Arrays

Time Complexity

Operation	Linked List (Singly)	Dynamic Array
Access	O(n)	O(1)
Search	O(n)	O(n)
Insertion at beginning	O(1)	O(n)
Insertion at end	O(1)	O(1)
Insertion at index	O(n)	O(n)
Deletion at beginning	O(1)	O(n)
Deletion at end	O(n)	O(1)
Deletion at index	O(n)	O(n)
Resizing (doubling)	N/A	O(n)
Traversal	O(n)	O(n)

Advantages and Disadvantages

Linked Lists:

- Advantages:
- Dynamic Size: Can easily grow and shrink as needed without reallocation or copying data.
- Efficient Insertions/Deletions: Efficient for insertions and deletions at the beginning or middle of the list.
- Memory Utilization: No pre-allocation of memory required; uses memory proportionally with the number of elements.
- Disadvantages:
- Sequential Access: No direct access to elements; must traverse from the beginning (O(n) time complexity for access).
- Extra Memory: Requires additional memory for pointers/references.
- Cache Performance: Poor cache performance due to non-contiguous memory allocation.

Dynamic Arrays:

- Advantages:
- Direct Access: Allows direct access to elements using index (O(1) time complexity for access).
- Cache Performance: Better cache performance due to contiguous memory allocation.
- Memory Overhead: Less memory overhead compared to linked lists (no pointers required).
- Disadvantages:
- Fixed Size: Must resize (often doubling) when capacity is exceeded, which involves copying all elements to a new array (O(n) time complexity for resizing).
- Inefficient Insertions/Deletions: Insertions and deletions in the middle or beginning require shifting elements (O(n) time complexity).
- Memory Reallocation: Resizing can lead to inefficient memory usage if the array is frequently resized.